Unsupervised Learning in Python

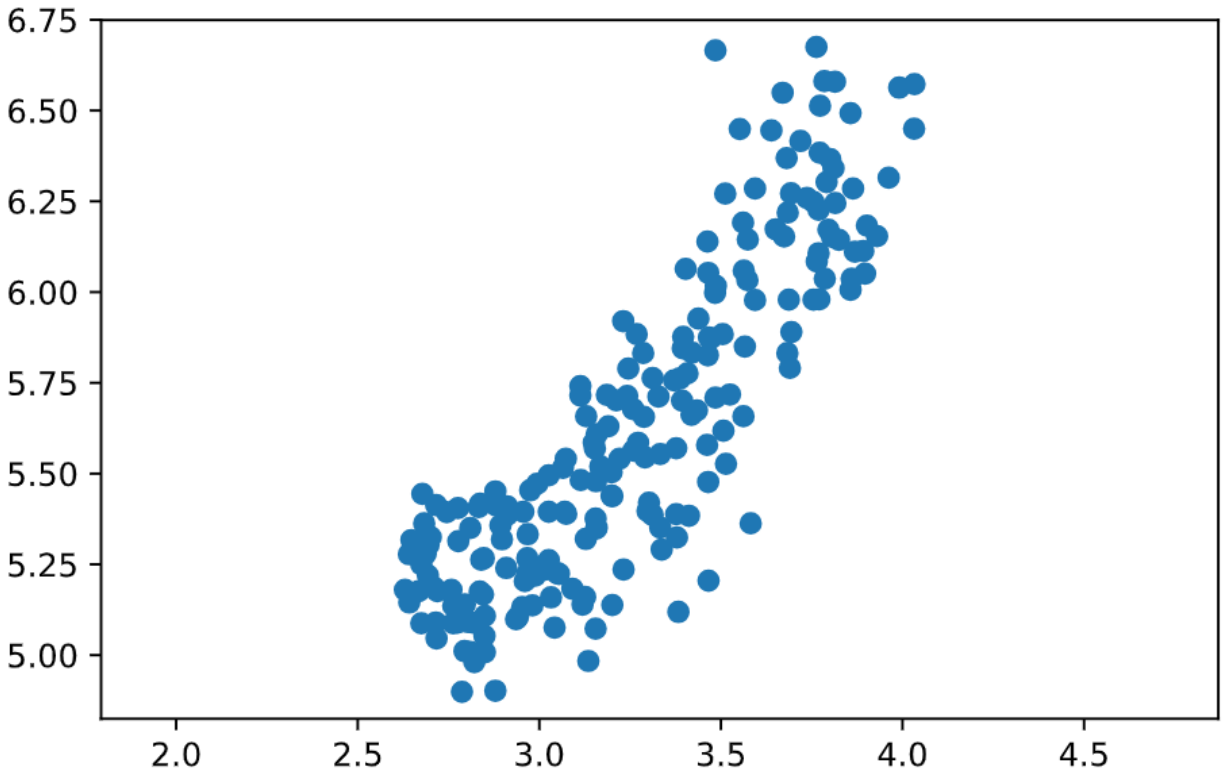
3). Decorrelating your data and dimension reduction:

a). Corelated data in nature

```
# Perform the necessary imports
import matplotlib.pyplot as plt
from scipy.stats import pearsonr
# Assign the 0th column of grains: width
width = grains[:,0]

# Assign the 1st column of grains: length
length = grains[:,1]

# Scatter plot width vs length
plt.scatter(width, length)
plt.axis('equal')
plt.show()

# Calculate the Pearson correlation
correlation, pvalue = pearsonr(width, length)
# Display the correlation
print(correlation)
```

<script.py> output:

0.860414937714

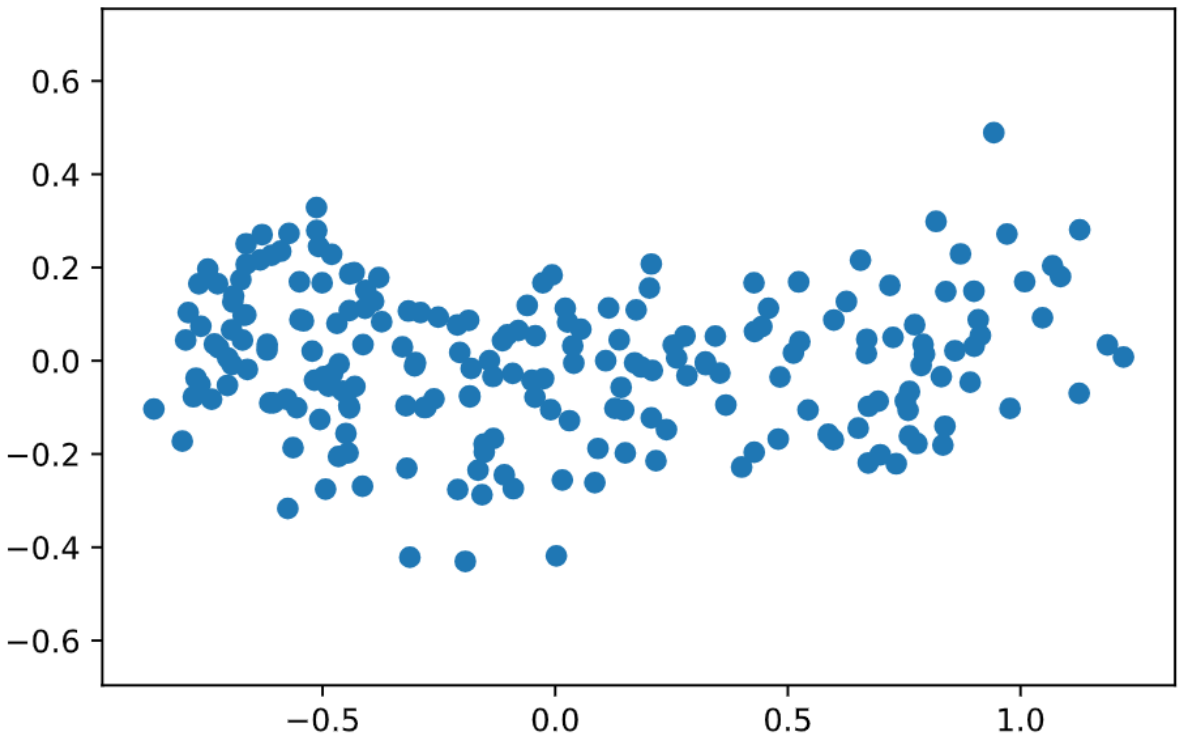

6.75
6.50
6.25
6.00
5.75
5.50
5.25
5.00
2.0
2.5
3.0
3.5
4.0
4.5

b). Decorrelating the grain measurements with PCA

```
# Import PCA
from sklearn.decomposition import PCA

# Create PCA instance: model
model = PCA()

# Apply the fit_transform method of model to grains: pca_features
pca_features = model.fit_transform(grains)

# Assign 0th column of pca_features: xs
xs = pca_features[:,0]

# Assign 1st column of pca_features: ys
ys = pca_features[:,1]

# Scatter plot xs vs ys
plt.scatter(xs, ys)
plt.axis('equal')
plt.show()
# Calculate the Pearson correlation of xs and ys
correlation, pvalue = pearsonr(xs, ys)
# Display the correlation
print(correlation)
```

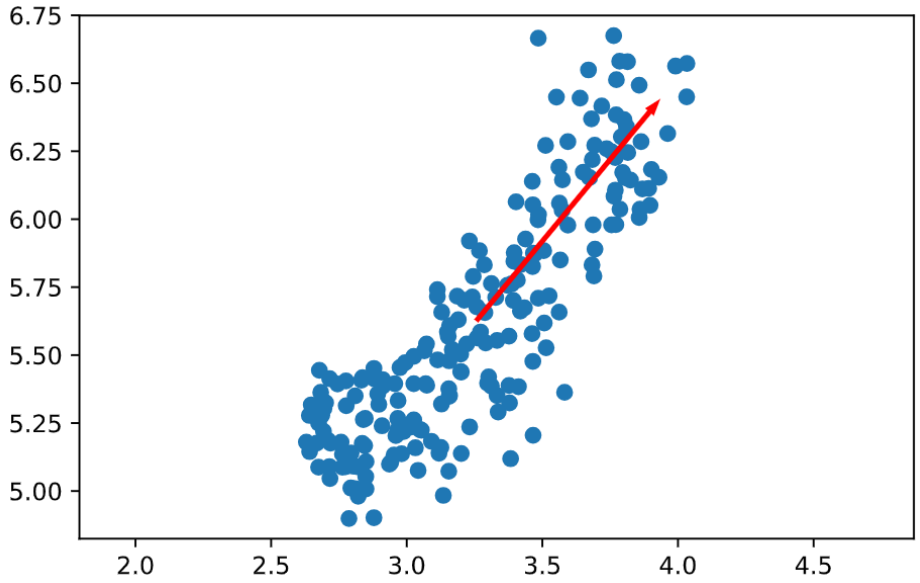
<script.py> output:

0.0


0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.5
0.0
0.5
1.0

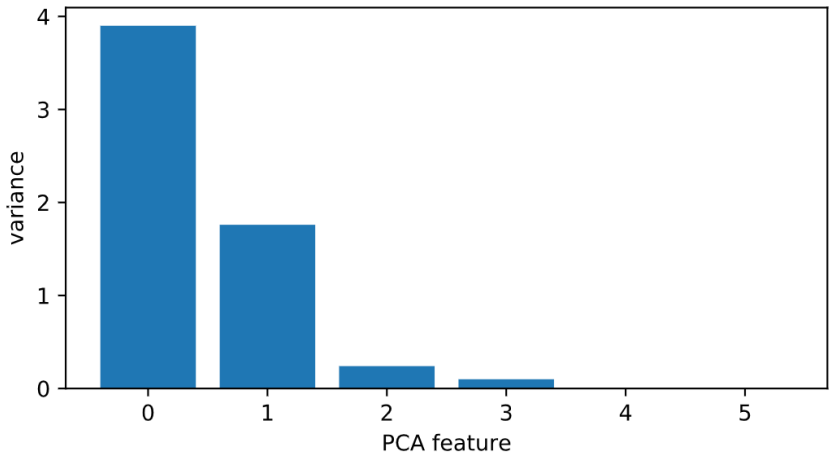
c). <u>The first Principal component</u>

```
# Make a scatter plot of the untransformed points
plt.scatter(grains[:,0], grains[:,1])

# Create a PCA instance: model
model = PCA()
# Fit model to points
model.fit(grains)

# Get the mean of the grain samples: mean
mean = model.mean_
# Get the first principal component: first_pc
first_pc = model.components_[0,:]

# Plot first_pc as an arrow, starting at mean
plt.arrow(mean[0], mean[1], first_pc[0], first_pc[1], color='red', width=0.01)

# Keep axes on same scale
plt.axis('equal')
plt.show()
```



d). Variance of the PCA features

```
# Perform the necessary imports
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
import matplotlib.pyplot as plt
# Create scaler: scaler
scaler = StandardScaler()

# Create a PCA instance: pca
pca = PCA()
# Create pipeline: pipeline
pipeline = make_pipeline(scaler, pca)

# Fit the pipeline to 'samples'
pipeline.fit(samples)
# Plot the explained variances
features = range(pca.n_components_)
plt.bar(features, pca.explained_variance_)
plt.xlabel('PCA feature')
plt.ylabel('variance')
plt.xticks(features)
plt.show()
```



e). Dimension reduction in the fish measurement

```
# Import PCA
from sklearn.decomposition import PCA

# Create a PCA model with 2 components: pca
pca = PCA(n_components=2)

# Fit the PCA instance to the scaled samples
pca.fit(scaled_samples)

# Transform the scaled samples: pca_features
pca_features = pca.transform(scaled_samples)

# Print the shape of pca_features
print(pca_features.shape)
```

<script.py> output:

(85, 2)

f). A tf-idf word-frequency array

```
# Import TfidfVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

# Create a TfidfVectorizer: tfidf
tfidf = TfidfVectorizer()

# Apply fit_transform to document: csr_mat
csr_mat = tfidf.fit_transform(documents)

# Print result of toarray() method
print(csr_mat.toarray())

# Get the words: words
words = tfidf.get_feature_names()

# Print words
print(words)
```

<script.py> output:

```
[[ 0.51785612  0.          0.          0.68091856  0.51785612  0.        ]
 [ 0.          0.          0.51785612  0.          0.51785612  0.68091856]
 [ 0.51785612  0.68091856  0.51785612  0.          0.          0.        ]]
['cats', 'chase', 'dogs', 'meow', 'say', 'woof']
```

g). Clustering Wikipedia part I

```
# Perform the necessary imports
from sklearn.decomposition import TruncatedSVD
from sklearn.cluster import KMeans
from sklearn.pipeline import make_pipeline

# Create a TruncatedSVD instance: svd
svd = TruncatedSVD(n_components=50)

# Create a KMeans instance: kmeans
kmeans = KMeans(n_clusters=6)

# Create a pipeline: pipeline
pipeline =make_pipeline (svd,kmeans)
```

h). <u>Clustering Wikipedia part II</u>

```
# Import pandas
import pandas as pd

# Fit the pipeline to articles
pipeline.fit(articles)

# Calculate the cluster labels: labels
labels = pipeline.predict(articles)

# Create a DataFrame aligning labels and titles: df
df = pd.DataFrame({'label': labels, 'article': titles})

# Display df sorted by cluster label
print(df.sort_values(['label']))
```

<script.py> output:

	article	label
59	Adam Levine	0
57	Red Hot Chili Peppers	0
56	Skrillex	0
55	Black Sabbath	0
54	Arctic Monkeys	0
53	Stevie Nicks	0
52	The Wanted	0
51	Nate Ruess	0
50	Chad Kroeger	0
58	Sepsis	0
0	HTTP 404	1
9	LinkedIn	1
1	Alexa Internet	1
2	Internet Explorer	1
3	HTTP cookie	1

4	Google Search	1
5	Tumblr	1
6	Hypertext Transfer Protocol	1
7	Social search	1
8	Firefox	1
48	Gabapentin	2
47	Fever	2
41	Hepatitis B	2
46	Prednisone	2
40	Tonsillitis	2
44	Gout	2
43	Leukemia	2
49	Lymphoma	2
45	Hepatitis C	2
42	Doxycycline	2
29	Jennifer Aniston	3
27	Dakota Fanning	3
26	Mila Kunis	3
24	Jessica Biel	3
23	Catherine Zeta-Jones	3
22	Denzel Washington	3
21	Michael Fassbender	3
20	Angelina Jolie	3
28	Anne Hathaway	3
25	Russell Crowe	3
11	Nationally Appropriate Mitigation Action	4
19	2007 United Nations Climate Change Conference	4
12	Nigel Lawson	4
13	Connie Hedegaard	4
10	Global warming	4
14	Climate change	4